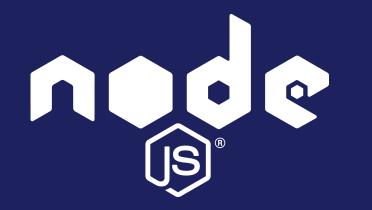
VENTURUS 4. I.C.

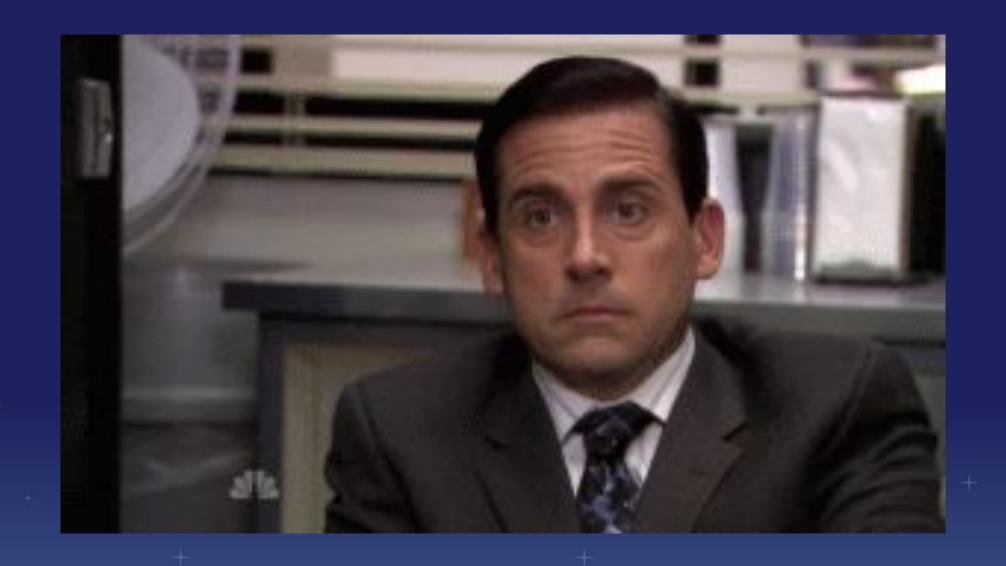


T = t

VENTURUS 4TECH







Segurança







Falhas de Segurança

- Nossa API está sem o mínimo de segurança;
- Qualquer pessoa com acesso ao endereço consegue cadastrar, editar e apagar uma vaga;



JWT - Json Web Token

- Padrão de mercado para comunicações em Json (RFC-7519);
- Formado por três seções:
 - Header: Define as informações sobre o tipo do token;
 - Payload: Contém as informações da entidade tratada (geralmente o usuário autenticado);
 - Signature: Concentração dos Hashes gerados a partir do Header e Payload.
- Apenas quem está de posse da chave pode, criar, alterar e validar o token.



JWT - Json Web Token

O resultado final é um token com três seções (header, payload, signature) separadas por "." (ponto).

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY30DkwIiwibmFtZSI6IkpvaG4gR
G9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab3
0RMHrHDcEfxjoYZgeFONFh7HgQ





Nossa aplicação hoje:







Exercício

- Instalar a dependência:
 - \$ npm i -S -E jsonwebtoken
- Criar uma coleção 'users' no firebase com os campos 'name', 'email' e 'password'
- Incluir pelo menos um documento nessa coleção
- DICA PARA MELHORIA: para realmente usar a senha seguramente, é
 preciso guardá-la encriptada. A sugestão é usar o método MD5 no node
 e no react (dá pra usar o mesmo pacote nos dois: npm md5). A senha
 só vai ser trafegada pela rede de forma encriptada e não aberta.





Exercício

- Criar um arquivo dentro de /app/routes
 - o Chamado login.js
 - o Implementar código
- Criar um diretório dentro da pasta config
 - o /security
 - o Criar um arquivo chamado tokenValidator.js
 - o Implementar código

 Atualizar os endpoint de jobs para que implementem o middleware(função) que criamos para fazer as validações



+Links e referências

- Autenticação JWT em Node.js
- <u>5 Passos para autenticar em node js com</u> <u>itw</u>
- MDN Web Docs window.localStorage
- Protected routes and authentication with React Router v4

